CHAPTER III

AN EXTENDED KTH-BEST APPROACH FOR INTEGER LINEAR BILEVEL PROGRAMING

3.1 Introduction

Kth-best algorithm is one of the most popular algorithms that have been proposed for the solution of linear bilevel programming problem. The basic idea in this algorithm is that the global optimal solution of the bilevel program is attained at an extreme point of the space constructed by the constraints of the leader and follower program. However, Shi, et al. (2005a), showed that it could not deal well with a linear bilevel programming when the constraints functions at the upper level are of the arbitrary linear form.

For \( x \in X \subseteq R^n, y \in Y \subseteq R^m, F : X \times Y \rightarrow R^1 \), and \( f : X \times Y \rightarrow R^1 \), a linear bilevel programming problem given by Bard (1998) is as follows

\[
\min_{x \in X} F(x, y) = c_1 x + d_1 y,
\]

\[
\text{s.t. } A_1 x + B_1 y \leq b_1,
\]

\[
\min_{y \in Y} f(x, y) = c_2 x + d_2 y,
\]

\[
\text{s.t. } A_2 x + B_2 y \leq b_2,
\]

where \( c_1, c_2 \in R^n, d_1, d_2 \in R^m, b_1 \in R^p, b_2 \in R^q, A_1 \in R^{p \times n} \).

Part of the results of this chapter is contained in Adhami et al. (2009a).
corresponding to (3.1.1) Bard gave the following basic definition for linear bilevel programming solution.

Definition 3.1.1

(a) Constraint region of the linear bilevel programming problem:

\[ S = \{ (x, y) : x \in X, y \in Y, A_1 x + B_1 y \leq b_1, A_2 x + B_2 y \leq b_2 \} \]

(b) Feasible set for the follower for each fixed \( x \in X \):

\[ S(x) = \{ y \in Y : B_2 y \leq b_2 - A_2 x \} \]

(c) Projection of \( S \) onto the leader’s decision space:

\[ S(X) = \{ x \in X : \exists y \in Y, A_1 x + B_1 y \leq b_1, A_2 x + B_2 y \leq b_2 \} \]

(d) Follower’s rational reaction set for \( x \in S(X) \):

\[ P(x) = \{ y \in Y : y \in \arg\min\{ f(x, \hat{y}) : \hat{y} \in S(x) \} \} \]

Where \( \arg\min\{ f(x, \hat{y}) : \hat{y} \in S(x) \} = \{ y \in S(x) : f(x, y) \leq f(x, \hat{y}), \hat{y} \in S(x) \} \)

(e) Inducible region:

\[ IR = \{ (x, y) : (x, y) \in S, y \in P(x) \} \]

Definition 3.1.2

\( (x^*, y^*) \) is said to be complete optimal solution, if and only if there exists \( (x^*, y^*) \in S \) such that \( F(x^*, y^*) \leq F(x, y) \) and \( f(x^*, y^*) \leq f(x, y) \) for all \( (x, y) \in S \).
The rational reaction set \( P(x) \) defines the response while the inducible region \( IR \) represents the set over which the leader may optimize his objective. Thus in terms of the above notations, the linear bilevel programming problem can be written as

\[
\min \{ F(x, y) : (x, y) \in IR \}
\]

These formulations and definitions are the foundation of linear bilevel programming theory and methods. But Shi et al. (2005a) showed that there exists a fatal deficiency for this theory as its performance is dependent on a linear form of the upper level constraint functions. Following the introduction, this chapter addresses the new definition for linear bilevel programming theory and an extended Kth-best approach to linear bilevel programming problem. The extended Kth-best approach is then further extended to solve the integer linear bilevel programming problem. At the end an example is given to illustrate the algorithm followed by a computer program for the introduced algorithm.

### 3.2 New Definition for Linear Bilevel Programming Problem and Extended Kth-Best Algorithm

The general formulation of a bilevel problem using common notations, is as follows

\[
\begin{align*}
\min_{x \in X} F(x, y) &= c_1 x + d_1 y, \\
\text{s.t. } A_1 x + B_1 y &\leq b_1, \\
\min_{y \in Y} f(x, y) &= c_2 x + d_2 y, \\
\text{s.t. } A_2 x + B_2 y &\leq b_2,
\end{align*}
\]

(3.2.1)
Corresponding to (3.2.1), Shi et al. gave the following new definition for linear bilevel programming solution:

**Definition 3.2.1**

(a) Constraint region of the linear bilevel programming problem:

\[ S = \{(x, y) : x \in X, y \in Y, A_1 x + B_1 y \leq b_1, A_2 x + B_2 y \leq b_2\} \]

The linear bilevel programming problem constraint region refers to all possible combinations of choices that the leader and the follower may make.

(b) Projection of \( S \) onto the leader's decision space:

\[ S(X) = \{x \in X : \exists y \in Y, A_1 x + B_1 y \leq b_1, A_2 x + B_2 y \leq b_2\} \]

Here it is suggested that the definition of bilevel programming model requires that the leader moves first by selecting an \( x \) in attempt to minimize its objecting subjecting to both upper and lower level constraints.

(c) Feasible set for the follower \( \forall x \in S(X) \):

\[ S(x) = \{y \in Y : (x, y) \in S\} \]

The follower's feasible region is affected by the leader's choice of \( x \), and the follower's allowable choices are the element of \( S \).

(d) Follower's rational reaction set for \( x \in S(X) \):

\[ P(x) = \{y \in Y : y \in \arg\min[f(x, \hat{y}) : \hat{y} \in S(x)]\} \]

The follower observes the leader's action and reacts by selecting \( y \) from his feasible set to minimize his objective function.
(e) Inducible region:

\[ IR = \{(x, y) : (x, y) \in S, y \in P(x)\} \]

Further it is proved that if \( S \) is nonempty and compact, there exists an optimal solution for a linear bilevel programming problem.

The basic idea of extended algorithm is that the extreme points of \( S \) are selected and checked if it is on inducible region \( IR \). If it is then, the current extreme point is the optimal solution. If not, select the next one and check.

More specifically, let \((x_{[1]}, y_{[1]}), \ldots, (x_{[N]}, y_{[N]})\) denote the \( N \) ordered extreme points to the linear programming problem

\[
\min \{c_i x + d_i y : (x, y) \in S\}, \tag{3.2.2}
\]

such that \( c_i x_{[i]} + d_i y_{[i]} \leq c_i x_{[i+1]} + d_i y_{[i+1]}, i = 1, \ldots, N - 1 \).

Let \( \overline{y} \) denote the optimal solution to the following problem

\[
\min(f(x_{[i]}, y) : y \in S(x_{[i]})). \tag{3.2.3}
\]

Now we only need to find the smallest \( i (i \in \{1, \ldots, N\}) \) under which \( y_{[i]} = \overline{y} \).

Let (3.2.3) be written as follows

\[
\min f(x, y) \tag{3.2.3}
\]

s.t. \( y \in S(x) \),

\[ x = x_{[i]} \]

From definition 3.2.1(a) and 3.2.1(c), we have
\[
\min f(x, y) = c_2 x + d_2 y \\
\text{s.t. } A_1 x + B_1 y \leq b_1,
\]
\[
A_2 x + B_2 y \leq b_2, \\
x = x_{[i]}, \\
y \geq 0.
\]

The solving is equivalent to select one ordered extreme point \((x_{[i]}, y_{[i]})\), then solve (3.2.4) to obtain the optimal solution \(\bar{y} = y_{[i]}\), \((x_{[i]}, y_{[i]})\) is the global optimum to (3.2.1). Otherwise, check the next extreme point.

### 3.3 The Extended Kth-Best Algorithm for Integer Linear Bilevel Programming Problem

According to the extended Kth-best approach for linear bilevel programming problem let us write the linear integer bilevel problem for leader in the following format

\[
\max F(x, y) = c_1 x + d_1 y \\
\text{s.t. } A_1 x + B_1 y \leq b_1,
\]
\[
A_2 x + B_2 y \leq b_2, \\
x \geq 0, \ y \geq 0 \text{ and integer}
\]

First we solve the linear programming relaxation for the above problem using simplex method. Let the solution be \((x^*, y^*)\). If the solution is non-integer we add the NAZ cut \(c_1 x + c_2 y \geq c_1 x^0 + c_2 y^0 = z^0\), which passes through \((x^0, y^0)\), an integer point inside the feasible region which has the
minimum positive difference from the objective function value at \((x_1^*, x_2^*)\).

\(z^0\) is the value of leader’s problem at \((x^0, y^0)\).

Let \((x^0, y^0)\) be defined as

\[(x^0, y^0) = (\alpha_1^{k_0}, \beta_2^{k_0}), \ldots, (\alpha_n^{k_0}, \beta_n^{k_0}) : K^0 \in S^0\]

where \(S^0\) = set of indices of \(2^n\) integer points in the surroundings of the non-integer solution \((x^*, y^*)\).

Now to find the integer optimum solution we add the A-T cut

\[
\sum_{j=1}^{n} (x_j + y_j) = \sum_{j=1}^{n} (\alpha_j^{k_0} + \beta_j^{k_0}) \text{ at } (x^0, y^0).
\]

Let \((x^*_1, y^*_1), \ldots, (x^*_N, y^*_N)\) denote the \(N\) ordered basic feasible solutions to the problem (3.2.2)

such that \(c_1x^*_i + d_1y^*_i \geq c_1x^*_{i+1} + d_1y^*_{i+1}, (i = 1, \ldots, N-1)\).

And for each of these values is associated an integer solution obtained after adding NAZ cut and A-T cut. Let these integer points be \((x^0_{[1]}, y^0_{[1]}), \ldots, (x^0_{[N]}, y^0_{[N]})\).

From definition 3.2.1(a) and 3.2.1(c), we can write the follower’s problem as follows

\[
\max f(x, y) = c_2 x + d_2 y
\]

s.t. \(A_1 x + B_1 y \leq b_1, \) \(A_2 x + B_2 y \leq b_2, \) \(x, y \geq 0\) \hspace{1cm} (3.3.2)
\[ x = x^0_{[i]}, \]
\[ y \geq 0 \text{ and integer.} \]

Let \( \bar{y} \) denote the integer optimal solution to the above problem.

We only need to find the smallest \( i (i \in \{1, \ldots, N\}) \) under which \( y^0_{[i]} = \bar{y} \).

The solving is equivalent to select one ordered integer point \( (x^0_{[i]}, y^0_{[i]}) \),
then solve (3.3.2) to obtain the optimal solution \( \bar{y} \). If \( \bar{y} = y^0_{[i]} \), then
\( (x^0_{[i]}, y^0_{[i]}) \) is the global optimum to (3.3.1). Otherwise, check the next
integer point.

The procedure can be summarized in the following steps:

**Step 1.** Set \( i = 1 \). Solve (3.2.2) with the simplex method. If the solution is
non integer then add NAZ cut and A-T cut to obtain integer optimum
solution as \( (x^0_{[i]}, y^0_{[i]}) \). Let
\( W = (x^0_{[i]}, y^0_{[i]}) \) and \( T = \phi \). Go to Step 2.

**Step 2.** Solve (3.3.2) for integer optimum solution using NAZ cut and A-
T cut. Let this solution be denoted by \( \tilde{y} \). If \( \tilde{y} = y^0_{[i]} \), stop; \( (x^0_{[i]}, y^0_{[i]}) \) is the
global optimum to integer linear bilevel programming problem.
Otherwise, go to Step 3.

**Step 3.** Let \( W_{[i]} \) denote the set of adjacent integer points such that
\[ c_i x^0_{[i]} + d_i y^0_{[i]} \geq c_i x^0_{[i+1]} + d_i y^0_{[i+1]}, \quad (i = 1, \ldots, N - 1). \]

Let \( T = T \cup (x^0_{[i]}, y^0_{[i]}) \) and \( W = (W \cup W_{[i]}) \cap T^c \). Go to step 4.
Step 4. Set $i = i + 1$ and choose $(x_{[i]}^{0}, y_{[i]}^{0})$ so that

$c_{1}x_{[i]}^{0} + d_{1}y_{[i]}^{0} = \max_{(x,y) \in \mathcal{R}} (c_{1}x + d_{1}y)$.

Go to step 2.

3.4 Numerical Illustration

Consider the following integer linear bilevel programming problem:

\[
\begin{align*}
\max_{x} F(x, y) &= 3x + 4y \\
\text{s.t.} & \\
3x + 2y &\leq 15 \\
x + 4y &\leq 10 \\
x &\geq 0, y &\geq 0 \text{ and integer}
\end{align*}
\]

\[
\begin{align*}
\max_{y \in \mathcal{Y}} f(x, y) &= 5x + 9y \\
\text{s.t.} & \\
2x + y &\leq 9 \\
x + 2y &\leq 7 \\
x &\geq 0, y &\geq 0 \text{ and integer}
\end{align*}
\]

According to the extended $K$th-best approach, above example can be written in the following format

\[
\begin{align*}
\max_{x} F(x, y) &= 3x + 4y \\
\text{s.t.} & \\
3x + 2y &\leq 15 \\
x + 4y &\leq 10 \\
2x + y &\leq 9
\end{align*}
\]
An Extended Kth-Best Approach....

\[ x + 2y \leq 7, \]
\[ x \geq 0, y \geq 0 \text{ and integer} \]

Solving the above problem by simplex method we get the non integer solution as \( x^* = 3.71, \ y^* = 1.57 \) and \( F(x^*, y^*) = 17.42 \)

We round off the non integer solution to the nearest four integer points as \((3, 1), (3, 2), (4, 1)\) and \((4, 2)\).

We are left with only one feasible point \((3, 1)\), which gives the minimum positive difference. Now the NAZ cut and A-T cut passing through the integer point \((3, 1)\) can be derived respectively as

\[ 3x + 4y \geq 13 \]

and

\[ x_1 + x_2 = 4 \]

Now solving the problem (3.4.2) with these additional constraints we obtain the integer optimum solution as

\[ (x_{[i]}, y_{[i]}) = (2, 2), \text{ and } F(x_{[i]}, y_{[i]}) = 14 \]

Let \((x_{[i]}, y_{[i]}) = (2, 2)\) the first best solution. Set \( W = \{(2, 2)\} \) and \( T = \phi \).

Go to step 2.

By (3.3.2), we have

\[ \max f(x, y) = 5x + 9y \]

s.t. \[ 3x + 2y \leq 15 \]
$x + 4y \leq 10$

$2x + y \leq 9$ \hspace{1cm} (3.4.3)

$x + 2y \leq 7$

$x = 2$

$x \geq 0, y \geq 0$ and integer

Solving the above problem, we have $\bar{y} = 2$. Since $\bar{y} = y_{[i]}$, we stop here.

The global solution to the problem is $(x_{[i]}^0, y_{[i]}^0) = (2, 2)$.

3.5 Computer Program

The program was run on Intel Centrino under Windows Vista operating system. The TURBO C++ version 5.0 was used for developing the program. The notations used for coefficients of objective function, constraints and variables are TS[], NC, NV respectively. The program is built using modular approach and the main modules that control the flow of the program are 'Data', 'Simplex' and Results. The 'Simplex' and 'Results' in turn uses different functions to evaluate the problem and applying NAZ and A-T cuts to obtain the optimal global solution for leader and follower objective functions.

The computer program is as follows:

```c
#include <stdio.h>
#include <math.h>
#define CMAX 10 //max. number of variables in objective function
#define VMAX 10 //max. number of constraints
int NC, NV, NOPTIMAL, P1, P2, XERR;
```
double TS[CMAX][VMAX];

void Data()
{
    double R1,R2;
    char R;
    int I,J;
    printf("INTEGER PROGRAMMING\n\n");
    printf("MAXIMIZE (Y/N) ? "); scanf("%c", &R);
    printf("NUMBER OF VARIABLES OF OBJECTIVE FUNCTION ? ");
    scanf("%d", &NV);
    printf("NUMBER OF CONSTRAINTS ? "); scanf("%d", &NC);
    if (R == 'Y' || R=='y')
        R1 = 1.0;
    else
        R1 = -1.0;
    printf("INPUT COEFFICIENTS OF OBJECTIVE FUNCTION:\n");
    for (J = 1; J<=NV; J++)
    {
        printf("#%d ? ", J); scanf("%lf", &R2);
        TS[1][J+1] = R2 * R1;
    }
    printf("Right hand side ? "); scanf("%lf", &R2);
    TS[1][1] = R2 * R1;
    for (I = 1; I<=NC; I++)
    {
        printf("CONSTRAINT #%d:\n", I);
        for (J = 1; J<=NV; J++)
        {
            printf("#%d ? ", J); scanf("%lf", &R2);
            TS[I + 1][J + 1] = -R2;
        }
    }
}
printf("Right hand side? "); scanf("%lf", &TS[I+1][1]);
}
printf("\n\nRESULTS:\n\n");
for(J=1; J<=NV; J++) TS[0][J+1] = J;
for(I=NV+1; I<=NV+NC; I++) TS[I-NV+1][0] = I;
}

void Pivot();
void Formula();
void Optimize();

void Simplex()
{

e10: Pivot();
Formula();
Optimize();
if (NOPTIMAL == 1) goto e10;
}

void Pivot()
{

double RAP,V,XMAX;
int I,J;
XMAX = 0.0;
for(J=2; J<=NV+1; J++)
{
if (TS[1][J] > 0.0 && TS[1][J] > XMAX)
{
XMAX = TS[1][J];
P2 = J;
}
}
RAP = 999999.0;
for (I=2; I<=NC+1; I++)
{
    if (TS[I][P2] >= 0.0) goto e10;
    V = fabs(TS[I][1] / TS[I][P2]);
    if (V < RAP)
    {
        RAP = V;
        P1 = I;
    }
e10:;
}
V = TS[0][P2]; TS[0][P2] = TS[P1][0]; TS[P1][0] = V;
}

void Formula()
{
    int I,J;
    for (I=1; I<=NC+1; I++)
    {
        if (I == P1) goto e70;
        for (J=1; J<=NV+1; J++)
        {
            if (J == P2) goto e60;
e60:;
        }
e70:;
    }
    TS[P1][P2] = 1.0 / TS[P1][P2];
    for (J=1; J<=NV+1; J++)
    {

if (J == P2) goto e100;
TS[P1][J] *= fabs(TS[P1][P2]);
e100:;
}
for (I=1; I<=NC+1; I++)
{
if (I == P1) goto e110;
TS[I][P2] *= TS[P1][P2];
e110:;
}

void Optimize()
{
int I,J;
for (I=2; I<=NC+1; I++)
if (TS[I][1] < 0.0) XERR = 1;
NOPTIMAL = 0;
if (XERR == 1) return;
for (J=2; J<=NV+1; J++)
if (TS[1][J] > 0.0) NOPTIMAL = 1;
}

void Results()
{
int I,J;
if (XERR == 0) goto e30;
printf(" NO SOLUTION.
"); goto e100;
e30:for (I=1; I<=NV; I++)
for (J=2; J<=NC+1; J++)
{
if (TS[J][0] != 1.0*1) goto e70;
printf(" VARIABLE #\%d: \%f\n", I, TS[J][1]);
)

} printf("OBJECTIVE FUNCTION: \%f\n", TS[1][1]);
printf(" The points after rounding of the solution the nearest integer are : (%s), (%s), (%s), (%s)", TSP[0], TSP[1], TSP[2], TSP[3]);
printf(" The best feasible point out of these is: (%d)", getFeasable(TSP));
printf("Now deriving the NAZ and A-T cut for this feasible point we get two added constraints : \%s , \%s", getConstraint(TSP));
printf("Press 'C' to continue solving the problem using new constraint.");
scanf("%c",&continue);
if(continue=='C')
{
    Data();
    Simplex();
    NAZ_AT_CUT_Results();
}
else
{
    printf("As the value of the variables for both leader and follower objective functions is same so using the bounded simplex we have the global solution as (x,y) = (%d,%d)", TS[0][0], TS[0][1]);
    
    e100:printf("\n");
}

void NAZ_AT_CUT_Results ()
{
    int I,J;
    if (XERR == 0) goto e30;
    printf(" NO SOLUTION.\n"); goto e100;
    for (I=1; I<=NV; I++)
        for (J=2; J<=NC+1; J++)
            for (L=1; L<=NC; L++)
                if (TSP[I][J][L]!=0) goto e100;
    e100:printf("\n");
}
An Extended Kth-Best Approach....

```
{ 
    if (TS[J][0] != 1.0*I) goto e70;
    printf(" VARIABLE #\%d: %f\n", I, TS[J][1]);
}
printf("\n OBJECTIVE FUNCTION: %f\n", TS[1][1]);
printf("\n Now to solve the problem for follower objective function with added constraints press 'C' ");
scanf("%c",&getVal);
if(getVal=='C')
{
    Data();
    Simplex();
    Results();
}
}

void main()
{
    Data();
    Simplex();
    Results();
}

The computer program solves the numerical problem as follows:

OUTPUT

INTEGER PROGRAMMING
MAXIMIZE(Y/N)  ? y
NUMBER OF VARIABLES OF OBJECTIVE FUNCTION ? 2
NUMBER OF CONSTRAINTS ? 4
INPUT COEFFICIENTS OF OBJECTIVE FUNCTION:
    #1? 3
    #2? 4
    Right hand side ? 0
CONSTRAINT #1:
    #1 ? 3
    #2 ? 2
    Right hand side ? 15
CONSTRAINT #2:
    #1 ? 1
    #2 ? 4
    Right hand side ? 10
CONSTRAINT #3:
    #1 ? 2
    #2 ? 1
    Right hand side ? 9
CONSTRAINT #4:
    #1 ? 1
    #2 ? 2
    Right hand side ? 7
RESULTS:
    VARIABLE #1 : 3.714286
    VARIABLE #2 : 1.571429
    OBJECTIVE FUNCTION: 17.428571
The points after rounding of the solution the nearest integer are: (3,1), (3,2), (4,1), (4,2)
The best feasible point out of these is: (3,1)

Now deriving the NAZ and A-T cut for this feasible point we get two added constraints:
3x+4y \geq 13
x+y=4

Press ‘C’ to continue solving the problem using new constraint. C

INTEGER PROGRAMMING

MAXIMIZE(Y/N) ? y

NUMBER OF VARIABLES OBJECTIVE FUNCTION ? 2
NUMBER OF CONSTRAINTS ? 6

INPUT COEFFICIENTS OF OBJECTIVE FUNCTION:

#1? 3
#2? 4
Right hand side ? 0

CONSTRAINT #1:

#1 ? 3
#2 ? 2
Right hand side ? 15

CONSTRAINT #2:

#1 ? 1
#2 ? 4
Right hand side ? 10

CONSTRAINT #3:

#1 ? 2
#2 ? 1
Right hand side ? 9
CONSTRAINT #4:
    #1 ? 1
    #2 ? 2
    Right hand side ? 7

CONSTRAINT #5:
    #1 ? 3
    #2 ? 4
    Right hand side ? 13

CONSTRAINT #6:
    #1 ? 1
    #2 ? 1
    Right hand side ? 4

RESULTS:
    VARIABLE #1 : 2
    VARIABLE #2 : 2

OBJECTIVE FUNCTION: 14

Now to solve the problem for follower objective function with added constraints press ‘C’

INTEGER PROGRAMMING

MAXIMIZE(Y/N) ? y

NUMBER OF VARIABLES OF OBJECTIVE FUNCTION ? 2

NUMBER OF CONSTRAINTS ? 5

INPUT COEFFICIENTS OF OBJECTIVE FUNCTION:
    #1? 5
    #2? 9
    Right hand side ? 0

CONSTRAINT #1:
    #1 ? 3
#2 ? 2  
Right hand side ? 15

CONSTRAINT #2:  
#1 ? 1  
#2 ? 4  
Right hand side ? 10

CONSTRAINT #3:  
#1 ? 2  
#2 ? 1  
Right hand side ? 9

CONSTRAINT #4:  
#1 ? 1  
#2 ? 2  
Right hand side ? 7

CONSTRAINT #5:  
#1 ? 1  
#2 ? 0  
Right hand side ? 2

RESULTS:  
VARIABLE #1 : 2.000000  
VARIABLE #2 : 2.000000

OBJECTIVE FUNCTION: 28.000000

As the value of the variables for both leader and follower objective functions is same so we have the optimal solution as \((x, y) = (2, 2)\).